Apprendre à penser les algorithmes
Concepts et exemple

Christian Blanvillain[1],
Université de Patras & Haute École Pédagogique de Lausanne

« L'informatique, c'est cool ! »
Marilou, 11 ans en 2019.
École active de Malagnou, Genève.

Avant-propos

Ce travail remarquable se situe dans le cadre de la thèse de l'auteur, codirigée par Messieurs les Professeurs Vassilis Komis de l'université de Patras (Grèce) et Bernard Baumberger de la Haute Ecole Pédagogique de Lausanne (Suisse).

En parcourant la toile, vous trouverez des compléments sur ses approches, tels que les évaluations humanistes : https://journals.openedition.org/ced/833 .

Nous connaissons Christian Blanvillain depuis fort longtemps. Il a été en poste à l'université de Provence à Marseille de 1997 à 2006. Nous rappelons ici son travail brillant et original qu'il a aussi présenté dans le bulletin ; comment couper un carré en trois carrés congruents :
http://biaa.eu/-upload/biaano86.htm . (N.D.L.R.)

Résumé

Nous présenterons un algorithme de résolution de problèmes algorithmiques. Cet algorithme est utile aux enseignants qui souhaitent faire apprendre aux élèves comment trouver des solutions à des problèmes algorithmiques. Un schéma des processus cognitifs mobilisés dans l'acte de conception d'algorithmes fait la synthèse de 42 fonctions cognitives. C'est un outil d'analyse pragmatique susceptible d'aider les enseignants à identifier les éventuelles lacunes cognitives des élèves en difficulté. De plus, pour aider les élèves en

[1] christian.blanvillain@hepl.ch

difficulté, 16 stratégies cognitives sont fournies. Elles interviennent durant les différentes phases du processus de résolution d'un problème algorithmique. Pour terminer, nous fournissons d'autres pistes pouvant expliquer les causes des difficultés des élèves. Nous concluons en proposant de créer une nouvelle discipline centrée sur le développement des intelligences de l'élève.

1. Introduction

Lorsque l'on souhaite enseigner l'informatique aux jeunes élèves, plusieurs questions se posent : peut-on se contenter de développer leur pensée informatique ou bien faut-il leur apprendre à programmer ? Doit-on leur enseigner un langage de programmation sur écran ou existe-t-il des solutions débranchées ? Finalement, l'essentiel n'est-il pas de leur apprendre à penser les algorithmes ?

Développer la pensée informatique sans leur apprendre à penser les algorithmes ni à coder et déboguer leur propre code, c'est passer à côté d'une opportunité extraordinaire de les initier à l'algorithmique. Développer l'intelligence algorithmique des élèves, dès leur plus jeune âge, permet de leur enseigner des stratégies cognitives pour la résolution de problèmes en cultivant un état d'esprit rigoureux et méthodique, qui leur serviront bien au-delà des exercices proposés et tout particulièrement dans les situations scolaires ou extrascolaires ou une approche intuitive ne suffit plus pour venir à bout des obstacles rencontrés. Leur apprendre à penser les algorithmes, c'est leur apprendre à développer leur intelligence créative, pratique et logico-mathématique (en référence à la conception triarchique de l'intelligence de Robert J. Sternberg, 1985). Les transferts possibles dans les autres disciplines scolaires et dans la vie quotidienne sont nombreux. Nous les préparerons ainsi à mieux appréhender le monde numérique dans lequel ils évoluent quotidiennement.

Nous avons ainsi conçu une formation complète pour initier les élèves à l'algorithmique et pour développer leurs stratégies cognitives. Le dispositif didactique et son support de cours sont disponibles en français et en anglais sur le site http://human-processor.xyz et sont partagés sous licence *creative commons* [CC-BY-NC-SA]. Le matériel proposé est constitué essentiellement de pièces en bois représentant les instructions. Elles permettent, par assemblage, d'écrire du code qu'il faudra ensuite interpréter à la main. Nous avons appelé ce

dispositif débranché *Human Processor!* (Blanvillain, 2020). Il est particulièrement adapté pour enseigner la conception et la programmation des algorithmes, sans distraire l'élève avec un environnement trop ludique qui favoriserait une approche par essaie-erreur plutôt qu'une approche réfléchie. Cet environnement tangible les force à vraiment construire leurs algorithmes, à penser leur code et à le vérifier manuellement. Ce dispositif s'inspire des exercices et du jeu électronique *Human Ressource Machine* mais en version simplifiée (il y a moins d'instructions) et totalement débranchée. Le langage du jeu *Human Ressource Machine* s'inspirant lui-même des travaux du Dr. Stuart Madnick, *Little Man Computer* (1965).

Notre dispositif est basé sur une machine notionnelle particulièrement simplifiée. Il est composé de deux types d'instruction : une instruction pour déplacer une information, une instruction (conditionnelle ou pas) pour sauter dans la séquence des instructions, des entrées/sorties et 3 mémoires. Ce minimalisme facilite son enseignement aux jeunes élèves dès l'âge de 10 ans. Malgré son extrême simplicité, notre dispositif est complet au sens de Turing. Il offre une plateforme réflexive permettant aux élèves d'échanger sur leurs stratégies de résolution de problème. L'enseignant alternera ainsi les moments de manipulation du dispositif, avec des moments de discussion autour des stratégies utilisées pour résoudre les problèmes proposés. Cette alternance va lui permettre d'aider les élèves en difficulté à développer un algorithme de résolution de problèmes algorithmiques. L'enseignant pourra alors leur apprendre à penser les algorithmes, en explicitant le processus présenté ci-dessous (chapitre 2), qui est une voie possible conduisant à la conception d'une solution algorithmique à un problème posé. Si un élève rencontre des difficultés d'apprentissage, la liste des fonctions cognitives (chapitre 3) viendra aider l'enseignant à identifier une éventuelle fonction cognitive ayant besoin d'être spécifiquement développée chez cet élève. Les stratégies cognitives (chapitre 4) leur permettant de bien comprendre le problème, de chercher une solution, d'apprendre à avoir de nouvelles idées, de valider leurs hypothèses et d'apprendre de leurs expériences. Nous concluons enfin (chapitre 5) en énonçant une liste de causes susceptibles d'expliquer les difficultés rencontrées par certains élèves pour apprendre à penser les algorithmes.

2. Algorithme de résolution de problèmes algorithmiques

Dans ce chapitre, nous décrivons un algorithme présentant comment l'élève pourrait s'y prendre pour résoudre des problèmes algorithmiques. L'algorithme est également fourni sous forme de diagramme en annexe. Cette théorie est issue d'une analyse introspective de notre propre processus de résolution de problème algorithmique, ainsi que de nombreuses lectures (voir bibliographie) :

- Jean-Yves Fournier (1999, 2ème partie) ;
- Pierre Vianin (2009, p. 72 à 116) ;
- Todd Lubart, Christophe Mouchiroud, Sylvie Tordjman, Franck Zenasni (2015, p. 117 et p. 122) ;
- Patrick Lemaire & André Didierjean (2018, chap. 8).

Dans le texte qui suit sont identifiées sept étapes mobilisant les fonctions cognitives présentées dans le prochain chapitre (figure 3). Elles sont écrites en gras et mises entre crochets.

Lecture et la compréhension de l'énoncé.

Pour résoudre un problème algorithmique, on commence par lire son énoncé. En première lecture, on peut facilement saturer notre mémoire de travail et ne pas comprendre le texte, ce n'est pas grave. Il ne faut pas paniquer et garder confiance en soi. Reprendre la lecture en essayant d'éliminer l'inutile et conserver les informations importantes pour identifier ce qui est demandé de faire ou de résoudre. Trier les informations fournies dans les consignes en cherchant les données de départ et les choses à produire **[1. Récolte]**. Cette étape est parfois nécessaire pour pouvoir être plus sélectif sur les informations pertinentes pour se représenter le travail à réaliser et les distinguer des informations qui le sont moins, qui posent juste le contexte. L'objectif est de ne conserver en mémoire de travail que les informations nécessaires pour que l'on puisse se créer une représentation mentale du problème à résoudre **[2. Assemblage]** en écartant les informations parasites. Il faut réussir à se représenter le problème à résoudre dans son ensemble : ce que l'on sait, ce que l'on cherche. Il s'agit de faire une synthèse de ce qui est demandé afin de pouvoir émettre une hypothèse heuristique sur la stratégie à utiliser et sur la démarche qui conduira à la solution **[3. Projection]**. Stratégie qu'il va falloir mentalement tester en une approche exploratoire rapide sur des données

simples, pour voir si elle est efficiente ou pas. L'exploitation de la représentation mentale du problème à résoudre que l'on s'est créée en lisant l'énoncé va permettre d'explorer plusieurs possibilités en éliminant rapidement les mauvaises idées, juste en testant les situations triviales.

Parmi les hypothèses faites sur les différentes stratégies de résolution possibles, conserver celle qui nous semble être la plus simple à mettre en œuvre, même si ce n'est pas forcément la plus élégante. Attention cependant à ne pas se précipiter sur la première idée qui vient à l'esprit ni se lancer dans l'élaboration d'une réponse sans avoir pris le temps de vérifier si l'idée semble valable ou pas. Tester cette hypothèse : appliquer le plan d'action prévu sur les données mémorisées, toujours en commençant par les situations les plus simples, car on cherche encore à valider notre hypothèse heuristique de résolution. Si notre hypothèse est correcte, on a un début de solution. Continuer alors d'analyser les autres données pour résoudre la suite du problème. Si notre hypothèse bloque sur certaines données et ne permet plus d'obtenir la réponse à la question posée, revoir l'hypothèse de départ et envisager d'utiliser une stratégie moins simple, mais plus précise. Un changement d'hypothèse peut aussi avoir lieu en cours de résolution : l'hypothèse initiale peut tout à fait fonctionner pour les cas simples, mais pas pour tous les cas. Les résultats obtenus ne sont pas forcément tous à remettre en cause : il ne faut pas obligatoirement jeter les réflexions faites et les conclusions partielles auxquelles on est arrivé. Une manière de chercher de nouvelles hypothèses est de faire des liens avec ce que l'on connait en cherchant des similitudes avec d'autres situations déjà rencontrées. Observer les ressemblances et les différences avec ce que l'on a déjà vécu par le passé peut nous aider à penser à une nouvelle manière de faire.

Lorsque l'on réussit à produire le résultat escompté dans les situations triviales, vérifier que la solution trouvée est consistante avec l'énoncé du problème. La compréhension partielle du problème va nous aider à mieux cerner ce qui est demandé et mieux comprendre ce qu'il faut faire pour pouvoir gérer les situations non triviales. Les réflexions initiales ne sont donc pas perdues. Elles étaient nécessaires pour arriver à ce stade du raisonnement. Elles nous permettent maintenant d'affiner nos réponses, de pouvoir reprendre l'ensemble de notre approche avec une conscience plus globale de ce qu'il faut faire en prenant en considération toutes les données du problème.

Le processus peut sembler coûteux en temps, mais il n'en est rien. Chaque étape permet de se familiariser davantage avec le problème et de mieux

appréhender les difficultés, les choses à faire et à ne pas faire, les informations essentielles et les informations moins cruciales. Cette expertise qui se construit en nous va nous permettre de nous approcher, mentalement et de manière itérative vers ce qui est attendu. Ce chemin intellectuel de compréhension est nécessaire pour se construire une représentation mentale fidèle du problème à résoudre. Parfois, le problème est suffisamment simple pour qu'il n'y ait pas besoin de faire de démarche particulière : on comprend tout de suite ce qui est demandé et ce qu'il faut faire. Mais parfois, le véritable problème n'apparait pas tout de suite et se découvre au fil du processus de lecture et de tentative de résolution. L'établissement d'une stratégie de résolution s'effectue conjointement au processus de compréhension du problème. Si l'on ne réussit pas à trouver une stratégie de résolution satisfaisante, c'est peut-être que l'on n'a pas assez bien exploré les données du problème. Une boucle se crée alors, entre la relecture pour une meilleure compréhension du problème, la recherche d'une nouvelle hypothèse heuristique et la vérification de l'hypothèse retenue. Nous appelons cette boucle le cycle d'élaboration d'une image mentale du problème (figure 1).

Résolution du problème

La phase de résolution commence lorsque l'on met en œuvre la stratégie de résolution imaginée sur l'ensemble des données du problème en envisageant les cas plus complexes. Concrètement, nous allons manipuler (ou plutôt *mentipuler* ?) les informations disponibles (données du problème) pour déplacer avec les yeux les données en faisant attention à ne pas oublier les nouveaux états du système que nous sommes en train de produire au travers de nos manipulations mentales **[4A. Réflexion]**.

À ce stade de la réflexion, il est très important de rester concentré sur la tâche : écarter tout évènement extérieur perturbateur, pour ne pas altérer notre état mental et la représentation que nous nous faisons du système. Notre mémoire de travail est fragile et éphémère. Si une perturbation que nous ne réussissons pas éviter à lieu (quelqu'un qui nous pose directement une question par exemple), ne pas hésiter à abandonner le raisonnement dans son ensemble et à recommencer depuis le début. Le nouvel état mental du système doit être fiable et dénué de toute erreur pour que nous puissions l'utiliser dans la suite du raisonnement, pour calculer les prochains états du système de manière itérative jusqu'à ce que nous atteignions la fin de notre algorithme de résolution associé

à notre stratégie et à nos hypothèses. Ces opérations mentales sont nécessaires et importantes pour que nous puissions nous assurer d'avancer, dans le raisonnement, vers une solution au problème posé. De temps en temps, si l'image mentale devient floue ou imprécise, reprendre rapidement les dernières étapes pour rafraichir notre mémoire de travail et éliminer les risques d'erreur de raisonnement dus aux imprécisions inhérentes à notre mémoire de travail.

Concrètement, réfléchir au code à écrire ou bien aux opérations à décrire pour élaborer un algorithme solution au problème posé, consiste à visualiser dans son esprit les différents états du modèle mental du système en cours de conception, puis de décrire les opérations qui vont faire évoluer cet état pour obtenir un état objectif qui est soit l'objectif final, soit une étape intermédiaire que l'on imagine nous rapprocher de l'objectif final. La représentation des états internes du modèle du système que nous programmons mentalement se fait en utilisant une image de l'algorithme en cours de construction. Cette image permet de visualiser les valeurs et leurs évolutions. Nous pouvons nous représenter les différentes opérations à effectuer comme si c'était nous qui devions physiquement réaliser ces opérations. En nous projetant à la place du système que nous sommes en train de programmer, nous mobilisons notre intelligence pratique : il suffit de faire appel au bon sens et de réfléchir à ce que nous devrions faire et pouvons faire pour obtenir ce que l'on souhaite, puis une fois que ces opérations sont conscientisées, de les décrire en langage naturel.

Souvent, une bonne stratégie de résolution consiste à découper le problème en étapes plus simples à résoudre et se concentrer sur chacune des étapes pour trouver une solution sans se préoccuper pour le moment des autres étapes. Mais lorsque l'on ne sait plus quoi faire et que l'on est bloqué, il faut observer avec un peu de recul l'état global du système dans notre tête, vérifier que nous avons bien exploité toutes les données à disposition, que nous avons bien exploré tout ce qu'il est possible de faire avec ces données, y compris les configurations de données improbables lorsque le problème est corsé. On peut donc faire un inventaire des données déjà utilisées et traitées, des résultats de réflexion intermédiaires obtenus et des données qu'il reste à traiter pour atteindre l'objectif qui nous est demandé. Si l'on ne sait vraiment plus quoi faire après avoir fait cet inventaire, on peut repasser en revue toutes les informations de l'énoncé du problème au lieu de se contenter de reparcourir uniquement les données conservées dans notre mémoire de travail. La recherche d'une éventuelle information oubliée dans l'énoncé risque de nuire à notre

représentation avancée de l'état du système et parfois, cet exercice nous obligera à reprendre notre raisonnement depuis le début pour intégrer la nouvelle information glanée dans l'énoncé. Cette gymnastique n'est pas aussi coûteuse en temps que ce que l'on pourrait croire, car refaire mentalement le chemin que nous venons juste de parcourir est assez rapide. Notre mémoire de travail va nous aider à aller bien plus vite la deuxième fois.

Si l'on est toujours bloqué dans l'étape de résolution du problème, avant d'abandonner notre hypothèse de travail, de tout jeter et de repartir sur une autre hypothèse plus complexe pour reprendre nos réflexions à partir de zéro, il convient de faire une pause. Relâcher la tension intellectuelle intérieure, calmer son esprit, laisser le cerveau se reposer quelques secondes en ne pensant à rien : une nouvelle idée pourrait alors jaillir spontanément nous permettant de nous débloquer **[4B. Idée]**. À ce stade du raisonnement, il est hyper important de ne pas se laisser distraire par les évènements extérieurs, ou par nos émotions intérieures : stress, doute, peur de ne pas y arriver, peur de se tromper, peur d'être en échec, autant d'émotions qui viennent perturber notre tranquillité et notre raisonnement. Toutes ces émotions doivent être mises en sourdine, pour laisser au cerveau la possibilité d'activer sa pensée créatrice et nous montrer une alternative que nous n'avions pas jusqu'alors envisagée. Mais pour que la magie opère, il faut accepter de ne pas savoir quoi faire, de ne pas avoir de réponse, d'être perdu et rempli de doute, sans sourciller, sans paniquer, sans arrêter de faire l'effort de chercher à faire des associations d'idées pour en générer de nouvelles ou bien de ne penser à rien pour laisser émerger spontanément de nouvelles idées.

Si toujours rien ne se passe, nous pouvons passer en revue des techniques de résolution connues, vues précédemment en nous demandant comment faire autrement, en tentant résoudre le problème en lui soumettant des principes déjà vus, des approches apprises, jusqu'à ce qu'une piste de solution apparaisse et que nous puissions repartir dans une nouvelle hypothèse de résolution, avec une stratégie adaptée. Nous pouvons également, à ce stade de la réflexion, nous aider de l'intelligence du groupe et demander aux autres élèves de partager leurs idées avec nous. Une idée fait émerger d'autres idées. Parfois, le chemin qu'emprunte la solution peut être assez détourné et c'est une idée à priori anodine qui pourrait nous mettre sur la voie d'une solution possible.

Expression de la solution

Lorsque l’on a terminé le processus de réflexion, que l’on a construit une solution à proposer, il faut vérifier qu'elle est bien juste **[5. Vérification]**. C'est une vérification plus lente qui s'opère par rapport à la vérification rapide de l'heuristique initiale, car ici on travaille avec une conscience complète de toutes les données du problème. Si la solution envisagée ne satisfait pas toutes les données du problème, chercher une nouvelle manière possible de résoudre le problème, adapter la stratégie de résolution et reprendre nos réflexions. Nous appelons cette boucle le cycle d'élaboration d'une image mentale de la solution (figure 1).

On peut être amené à élaborer des explications plus ou moins complexes pour présenter nos idées et notre résultat **[6. Expression]** à d'autres personnes. Lorsque l'on exprime notre solution, bien s’assurer que nos idées soient structurées, concises et compréhensibles par nos interlocuteurs. Enfin, lorsque tout est fini, repenser à ce qui a été réalisé. Se remémorer la stratégie qui nous a conduits à la résolution du problème posé et essayer alors d’identifier des situations similaires où cette stratégie serait à même de fonctionner pour favoriser les transferts, ou bien des situations déjà vécues dans lesquelles nous aurions déjà exploité cette stratégie de résolution. Ce faisant, nous créons des liens entre des expériences distinctes. Ces prises de conscience viennent alors enrichir notre propre expérience et nous préparent à mieux affronter des situations inédites.

La figure 2 ajoute à la représentation des deux cycles d’élaboration des images mentales du problème et de la solution, l’indication des différentes étapes mobilisant les fonctions cognitives. Dans le prochain chapitre, nous présentons la liste de ces fonctions cognitives utiles dans l’acte de concevoir une solution algorithmique.

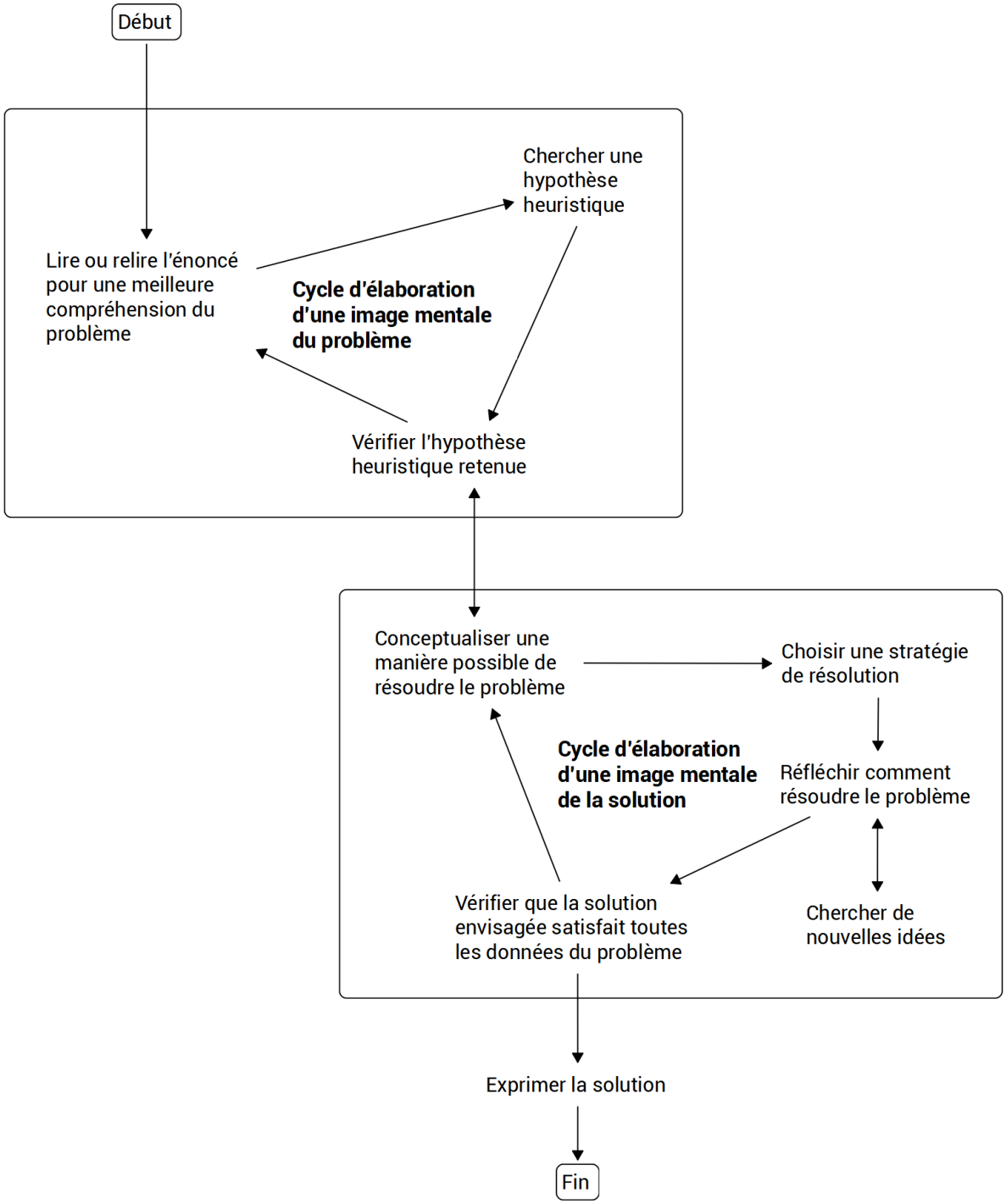


Figure 1. Les deux cycles d'élaboration des images mentales

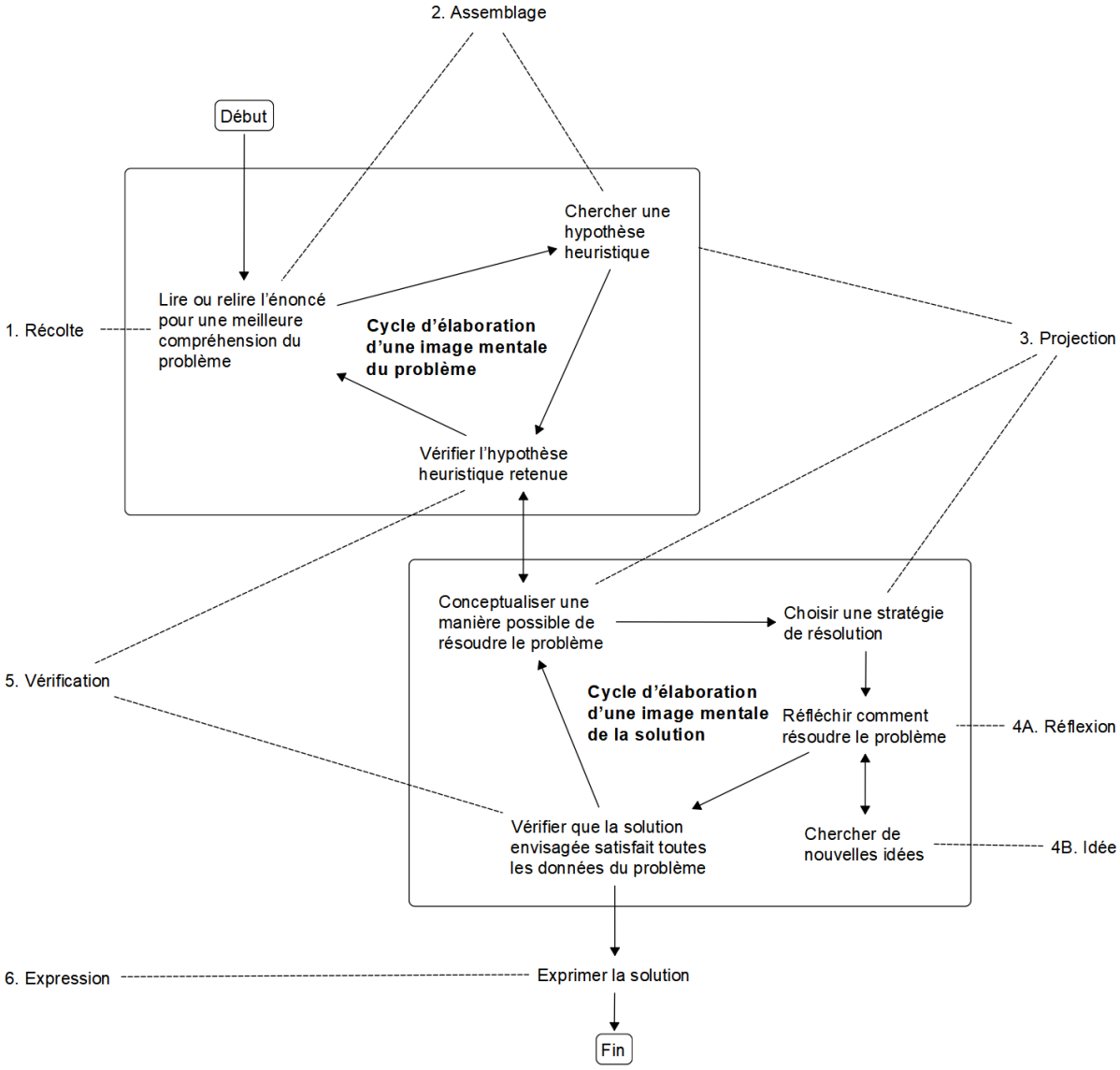


Figure 2. Liens entre les étapes mobilisant les fonctions cognitives
et les deux cycles d'élaboration des images mentales

3. Fonctions cognitives

Les fonctions cognitives listées dans cette section sont issues des travaux de plusieurs chercheurs (voir bibliographie) :

- Christine Mayer qui reprend Reuven Feurenstein dans son *Manuel de métapédagogie* (2005) ;
- Pierre Vianin, tableau récapitulatif dans *Neuroscience cognitives et pédagogie spécialisée* (2010) ;
- Todd Lubart, Franck Zenasni, Baptiste Barbot dans *Créativité en éducation et formation* (2016) ;
- Todd Lubart, Christophe Mouchiroud, Sylvie Tordjman, Franck Zenasni dans la *Psychologie de la créativité* (2015).

Le lien entre l'algorithme de résolution de problèmes algorithmique et les fonctions cognitives listées ci-dessous est réalisé par un mot clé en gras et entre crochets dans le texte du chapitre précédent. Il identifie l'étape du processus durant lequel nous considérons que les fonctions cognitives associées sont plus particulièrement susceptibles d'être mobilisées par l'élève. Nous avons identifié 7 étapes, regroupées dans les 3 phases de l'algorithme : (a) lecture et compréhension de l'énoncé, (b) résolution du problème et (c) expression de la solution. Elles sont représentées dans le diagramme ci-dessous (figure 3).

Ce travail de synthèse est un outil utile pour l'enseignant qui chercherait à faire une analyse clinique des éventuelles causes de blocage chez l'élève en difficulté. Cette liste offre aux enseignants de nouvelles lunettes de compréhension sur les principaux mécanismes cognitifs potentiellement mis en œuvre par l'élève dans les étapes de conception d'une solution à un problème algorithmique. Elle ouvre de nouvelles pistes de travail de remédiation ciblé, lorsqu'une éventuelle défaillance cognitive est suspectée ou identifiée. Loin d'être exhaustive, cette liste pragmatique se concentre sur les 42 fonctions cognitives que nous considérons être les plus pertinentes dans la tâche de résolution de problèmes algorithmiques. Il se peut que les raisons du blocage de l'élève soient ailleurs, mais nous espérons que dans la majorité des situations, l'enseignant trouvera dans ce répertoire une des origines possibles à ses difficultés d'apprentissage. Les fonctions cognitives sont listées avec un exemple inspiré de l'environnement de formation accompagnant le dispositif didactique *human processor!* qui est disponible en libre accès sur internet (voir introduction).

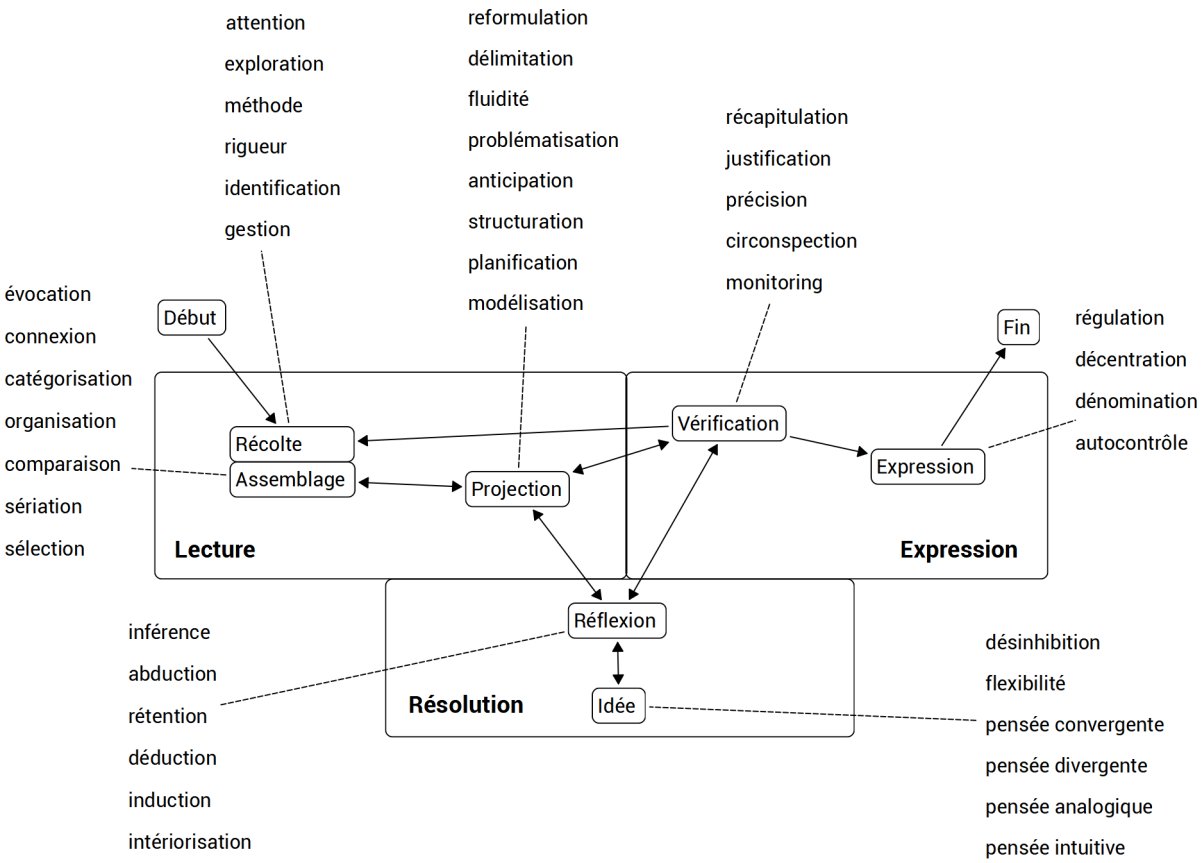


Figure 3. Les trois phases de l'algorithme de résolution de problèmes algorithmiques en lien avec la liste des 42 fonctions cognitives susceptibles d'être mobilisées dans les 7 étapes correspondantes.

[1.Récolte] La récolte des données

1.1 Attention	**1.2 Exploration**	**1.3 Méthode**
Orienter ses sens délibérément et intentionnellement vers une source d'information. Focalise ton attention sur ce que je vois et ce que j'entends. *Exemple : regarde le maître dans les yeux lorsque celui-ci lui parle. Est capable d'écouter une explication jusqu'au bout.*	Effectuer une observation systématique de la tâche. *Exemple : prends le temps de regarder toutes les pièces du jeu, même celles qui ne sont pas encore présentées.*	Examiner tout. Explorer systématiquement les informations disponibles pour ne pas sauter ou oublier quelque chose d'important ou pour ne pas répéter une information. *Exemple : prends le temps de bien lire l'énoncé. S'aider des exemples fournis.*
Mayer (2005) Vianin (2010)	Vianin (2010)	Mayer (2005)

1.4 Rigueur	**1.5 Identification**	**1.6 Gestion**
Être précis et exact dans le rassemblement des données. *Exemple : synchronise l'accumulateur dans le code avec l'état du débogueur et garde-le synchronisé tout au long du débogage.*	Identifier le type de problème. Déterminer les caractéristiques et les attributs d'un objet en identifiant les ressemblances. Faire des liens avec les expériences passées. *Exemple : comprends facilement la différence entre un jump et un if jump.*	Prendre plusieurs sources d'informations à la fois. Être attentif à plusieurs choses en même temps. *Exemple : exécute ton code, déplace l'accumulateur, note l'état de la mémoire dans le débogueur, en même temps, sans décalage.*
Mayer (2005)	Vianin (2010)	Mayer (2005)

[2.Assemblage] L’assemblage des données

2.1 Évocation	**2.2 Connexion**	**2.3 Catégorisation**	**2.4 Organisation**
Donner un sens à l’information perçue en évoquant une représentation mentale. *Exemple : arrive à te représenter mentalement l’architecture de Von Neumann, la position de l’accumulateur et la valeur qu’il contient à l’instant* t*.*	Établir des liens, des relations, des associations entre les objets, les événements, les expériences. Avoir une compréhension de la réalité non fragmentée. *Exemple : fais des liens entre les instructions des exercices du tutoriel et les instructions dont tu as besoin dans les exercices suivants.*	Extraire les caractéristiques communes à un ensemble d’objets et constitue des classes sur la base de ces similitudes. *Exemple : range les pièces correctement dans leur boite.*	Synthétiser les informations en les organisant en un tout cohérent et en créant des unités de sens. *Exemple : utilise le débogueur convenablement lorsque c’est nécessaire, sans sauter d’étapes et sans se tromper dans les étapes.*
Vianin (2010)	Mayer (2005) Vianin (2010)	Vianin (2010)	Vianin (2010)

2.5 Comparaison	**2.6 Sériation**	**2.7 Sélection**
Distinguer les objets entre eux. Déterminer leurs ressemblances et leurs différences. *Exemple : fais bien la différence entre l’input et l’output : n’écrit pas sur l’input et ne lit pas sur l’output.*	Classer les éléments dans l’ordre chronologique (séquence temporelle). *Exemple : écris les opérations de ton code en séquences chronologiques.*	Trier les informations pertinentes et éliminer les informations inutiles. *Exemple : parviens à retenir les informations importantes dans les énoncés des problèmes.*
Mayer (2005) Vianin (2010)	Vianin (2010)	Mayer (2005) Vianin (2010)

[3.Projection] La projection d'une stratégie de résolution

3.1 Reformulation	**3.2 Délimitation**	**3.3 Fluidité**	**3.4 Problématisation**
Décrire, expliquer, résumer ce qui a été compris. *Exemple : sois capable d'énoncer de manière intelligible l'exercice que tu es en train de tenter de résoudre.*	Choisir le cadre, le contexte et les moyens à disposition pour résoudre le problème. *Exemple : identifie parmi les pièces et instructions disponibles, lesquelles vont être susceptibles de réaliser les opérations à programmer. Écarte les pièces inutiles.*	Ouverture d'esprit. Tolérance à l'ambiguïté. *Exemple : sois capable d'avancer dans la résolution de l'exercice en acceptant le doute lié à l'ordre des opérations pour l'exercice du* soustracteur*.*	Définir le problème. Ce que l'on me demande. Ce que j'ai besoin de faire. *Exemple : réussis à comprendre ce qu'il faut faire et à te mettre en condition pour le résoudre.*
Vianin (2010)	Mayer (2005)	Lubart (2016) p.70	Mayer (2005)

3.5 Anticipation	**3.6 Structuration**	**3.7 Planification**	**3.8 Modélisation**
Fixer un but et envisager le déroulement et les éventuelles difficultés qui pourraient se présenter. *Exemple : exprime oralement ce qui doit être fait ou comment le faire avant de le faire.*	Structurer sa tâche en étapes et en sous-étapes. *Exemple : parviens à résoudre* l'octoplicateur *en trois additions.*	Orienter l'action vers le but fixé et organiser la chronologie des étapes. *Exemple : enchaîne de toi-même les exercices du tutoriel sans que l'enseignant ait besoin de lui dire quoi faire une fois un exercice terminé.*	Se construire un modèle mental de la chose sur laquelle on est en train de travailler. *Exemple : assemble les images mentales en un système dynamique cohérent manipulable mentalement.*
Vianin (2010)	Vianin (2010)	Mayer (2005) Vianin (2010)	

[4A.Réflexion] L’élaboration d’une réflexion

4A.1 Inférence	**4A.2 Abduction**	**4A.2 Rétention**	**4A.4 Déduction**
Déduire logiquement une proposition en réalisant des liens avec des propositions préalables tenues pour vraies. *Exemple : comprends les limites des algorithmes de multiplication proposés et déduis qu’une généralisation ne peut pas être réalisée avec les pièces du jeu, contrairement à la méthode bourrin.*	Formuler des hypothèses intuitives. Envisager d’autres possibilités tout en cherchant une conclusion concordante aux observations et en éliminant les solutions improbables. *Exemple : réussis l’exercice optionnel multiplier par 7 en changeant de stratégie et en utilisant une soustraction plutôt que seulement des additions.*	Retenir toutes les informations utiles dans sa tête pour pouvoir les utiliser simultanément. À une vision mentale large. *Exemple : parviens à mémoriser facilement toutes les définitions des différents composants de l’architecture de Von Neumann.*	Tirer une conclusion de la réflexion. Rapport de cause à effet. Si… alors… Chercher les causes, les conséquences. *Exemple : comprends que pour l’exercice* conservateur de zéro *il manque l’instruction différent de zéro au langage et que si l’on n’a pas cette instruction on peut quand même s’en sortir avec un saut judicieux.*
Mayer (2005) Vianin (2010)	Vianin (2010)	Mayer (2005)	Vianin (2010)

4A.5 Induction	**4A.6 Intériorisation**
Tirer des conclusions générales. Remonte des faits aux lois ou aux règles. *Exemple : comprends la généralisation de l’algorithme de la multiplication égyptienne. Se demander comment manipuler les bits avec* hp!	Travailler dans sa tête uniquement. À une bonne image mentale de ce qu’il s’agit de faire. Déplace les choses dans sa tête, avec ses yeux. *Exemple : réfléchis aux solutions du problème sans manipuler les pièces du dispositif.*
Vianin (2010)	Mayer (2005)

[4B.Idée] La recherche d'une idée

4B.1 Désinhibition	**4B.2 Flexibilité**	**4B.3 Pensée convergente**
Accepter la prise de risque. Propension à oser. *Exemple : souhaite tenter de faire les derniers exercices, alors que les précédents ne sont pas encore tous terminés.*	Ne pas se bloquer. Ne pas paniquer. Chercher spontanément une autre manière de faire. *Exemple : ne te met pas à jouer lorsqu'il rencontre une difficulté. Envisage d'autres manières de faire, puis expérimente les.*	Réaliser des combinaisons sélectives. Synthétiser différents éléments en un seul et même système par combinaison de deux structures complexes. *Exemple : associe deux idées distinctes pour en générer une nouvelle.*
Lubart (2016) p. 70	Mayer (2005) Lubart (2016) p.70 Lubart (2015) Chap. 2.7	Lubart (2016) p.70

4B.4 Pensée divergente	**4B.5 Pensée analogique**	**4B.6 Pensée intuitive**
Lorsque la pensée convergente est en échec, penser "out of the box" permet d'envisager de nouvelles alternatives et, parfois, de trouver des solutions originales. *Exemple : propose des solutions originales aux problèmes posés.*	Utiliser de comparaisons sélectives par le biais des analogies et des métaphores pour trouver de nouvelles idées. *Exemple : fais des analogies avec les mathématiques comme la représentation géométrique des nombres sur une droite dans le* soustracteur*.*	S'appuyer sur ses propres expériences, ses propres émotions et sentiments. *Exemple : manifeste intuitivement la possibilité de programmer* l'octoplicateur *en réutilisant les calculs intermédiaires pour économiser des opérations.*
Lubart (2016) p.70 Lubart (2015) Chap. 2.5	Lubart (2016) p.70	Lubart (2016) p.70

[5.Vérification] La vérification d'une réponse

5.1 Récapitulation	**5.2 Justification**	**5.3 Précision**
Vérifier, compter, inventorier ou résumer, tout ce à quoi j'ai pensé. Faire la synthèse avant de répondre. *Exemple : évite de proposer des solutions partiellement réfléchies. À envisager tous les cas avant de prendre la parole pour proposer son idée.*	Utiliser un raisonnement logique pour prouver les choses ou pour vérifier leur exactitude. *Exemple : appuie son argumentation sur une preuve mettant en œuvre le déplacement des données dans l'architecture de Von Neumann pour convaincre son pair de l'exactitude de sa réflexion.*	Faire attention à l'exactitude des éléments de réponse donnés. *Exemple : ne te trompe pas dans le nom des mémoires utilisées pour calculer l'exercice du* décaplicateur*.*
Mayer (2005)	Mayer (2005)	Mayer (2005)

5.4 Circonspection	**5.5 Monitoring**
Ne pas être impulsif. Ne se pas précipiter. Ne pas dire la première chose qui lui passe par la tête. *Exemple : pense à gérer le cas des nombres négatifs dans l'exercice* à rebours*.*	Surveillance de l'action. Contrôler l'efficacité des stratégies choisies et garder le but en tête. *Exemple : utilise systématiquement le débogueur avant de demander à l'enseignant si le code est correct.*
Mayer (2005)	Vianin (2010)

[6.Expression] L’expression d’une réponse ou d’une solution

6.1 Régulation	**6.2 Décentration**	**6.3 Dénomination**	**6.4 Autocontrôle**
Ajuster ou réorienter les stratégies utilisées. *Exemple : réalise tes erreurs et envisage par lui-même des solutions alternatives, lorsqu’il communique avec ses pairs ou l’enseignant.*	Traduire la réflexion dans un langage écrit ou oral compréhensible par autrui. Se mettre à la place de l’autre pour être sûr d’être compris. Communiquer de manière non égocentrique. *Exemple : les explications prodiguées sont claires.*	Connaitre et utiliser le bon nom des choses. Avoir le vocabulaire pour décrire les expériences et pour mieux les retenir. *Exemple : évite d’utiliser truc, machin, bidule pour parler de son code.*	Comparer le résultat obtenu avec le but recherché. *Exemple : préoccupe-toi de savoir si ses pairs ou l’enseignant ont bien compris l’idée qui a été présentée.*
Vianin (2010)	Mayer (2005) Vianin (2010)	Mayer (2005)	Vianin (2010)

4. Stratégies cognitives

Les stratégies cognitives listées dans cette section reprennent une bonne partie des idées présentées dans l'algorithme de résolution de problèmes algorithmiques et les reformulent pour les rendre accessibles aux élèves. Certaines de ces stratégies sont directement inspirées de Christine Mayer (2005). D'autres ont été élaborées à partir des discussions et réflexions de groupe faites en classe lors des précédentes itérations de la formation qui accompagne le dispositif didactique *human processor!*. C'est le fruit d'un travail empirique de médiation (Chappaz, 1995, p. 13) de la part de l'enseignant réalisant, avec les élèves, une analyse réflexive et métacognitive (Chappaz, 1995, p. 17-19) des stratégies concrètes et opérationnelles utilisées. Elles sont utiles pour aider les élèves en difficulté à trouver les solutions aux problèmes algorithmiques proposés.

Ces 16 stratégies sont utiles à 4 moments distincts du processus de résolution d'un problème algorithmique : avant de se mettre au travail, pendant la recherche d'une solution, au moment où l'on est bloqué et enfin, lorsque le travail est terminé, dans une étape d'analyse réflexive qui permet de capitaliser les savoirs acquis, enrichir l'expérience, améliorer la prise de conscience et les apprentissages effectués.

Les stratégies utiles pour apprendre à se mettre au travail :

[CAPABLE] *Je suis capable de le faire.*	**[FOCUS]** *Je me concentre pour réussir.*	**[PAUSÉ]** *Je résiste à donner la première réponse sans vérifier.*	**[PENSE]** *Je pense avant d'agir.*
Qu'est-ce que je dois faire ? Est-ce que j'ai bien toutes les données ? C'est ok de ne pas y arriver du premier coup ou de faire des erreurs : garde confiance en toi. Si c'était trop facile, tu n'apprendrais rien.	Est-ce que tu as besoin qu'on te résume ce qu'il faut faire ? Focalise ton attention sur la tâche que tu es en train d'accomplir. Tu penseras au reste plus tard. Il y a un temps pour apprendre et un temps pour se s'amuser.	Est-ce que tu as pris le temps avant de donner ta réponse ? Vérifie bien l'exactitude de ton idée. Parfois la vérité se cache ailleurs. Une approche critique et réfléchie permet d'éviter les pièges.	Comment ne pas répondre au hasard ? Demande-toi, si tu fais ça, alors qu'est-ce qui se passe ? Tu exploreras très rapidement beaucoup plus de possibilités en raisonnant dans ta tête qu'en essayant pour de vrai.

Les stratégies pour apprendre à résoudre un problème :

[LIENS] *Je fais des liens avec ce que je connais.*	**[IMAGE]** *Je me construis une image mentale du problème.*	**[MENTAL]** *Je manipule mon image mentale pour résoudre le problème.*	**[SIMPLE]** *Je cherche à simplifier le problème.*
Qu'est-ce qui est pareil ? Qu'est-ce qui est différent ? Cherche des similitudes avec ce que tu sais déjà et fais une hypothèse "c'est comme si…". Observe les différences et essaye de voir si ton intuition est juste.	Qu'est-ce qui est utile ? Qu'est-ce qui ne l'est pas? Qu'est-ce que je garde dans ma tête ? Représente-toi visuellement ce qu'il faut faire avant de commencer, ça va t'aider à mieux réfléchir.	Utilise l'image que tu as construite dans ta tête pour chercher une solution avant de l'essayer pour de vrai. Déplace les choses avec tes yeux pour vérifier tes idées.	Découpe le problème en plus petits problèmes ou bien résous un problème semblable, mais plus simple. Qu'est-ce qui est constant ? Qu'est-ce qui est variable ? Essaye de généraliser ta solution.

Les stratégies pour apprendre à se débloquer :

[BACKTRACK] *J'explore une autre piste.*	**[FLEXIF]** *Je change de chemin.*	**[CRÉATIF]** *Je trouve de nouvelles idées.*	**[GROUPE]** *J'utilise l'intelligence du groupe.*
Lorsqu'une piste de réflexion ne mène à rien, faire marche arrière et envisager d'autres variations alternatives partout où un choix a été fait.	Quel est le contexte ? Quelles sont toutes les possibilités ? Si tu es bloqué, cherche à changer de tâche ou de stratégie. Demande-toi comment faire totalement autrement.	Ferme les yeux et évacue toutes les pensées parasites en utilisant ton attention pour faire taire la petite voix dans ta tête. Des idées vont émerger d'elles-mêmes.	Bloqué ? Cherche de l'aide et lorsque tu as trouvé, regarde si tu peux à ton tour aider les autres à trouver par eux-mêmes. Demande-toi si les autres vont comprendre ce que tu as fait ou ce que tu dis.

Les stratégies réflexives pour apprendre à mieux apprendre :

[CALME] *Quand la tempête est passée, j'apprends à baisser les tours.*	**[RÉFLEXIF]** *Je réfléchis aux chemins pris pour apprendre.*	**[OPTIMUM]** *Je réfléchis plus pour travailler moins et apprendre mieux.*	**[TRANSFERT]** *Je réfléchis aux applications futures*
Lorsque je ne trouve pas, que je suis bloqué, je ne dois pas stresser ni paniquer. Je dois juste m'habituer à accepter le doute et l'incertitude. C'est la condition pour laisser émerger de nouvelles idées.	Quelles sont les étapes que tu as suivies ? Juste après avoir résolu un problème, pose-toi la question de ce qui t'a permis de résoudre le problème pour t'en souvenir plus tard.	Qu'est-ce que tu as vu ou entendu dans ta tête ? Comment as-tu su que c'était correct ? Dirige ton attention sur ta manière d'apprendre, pour apprendre mieux et avec moins d'efforts.	Quel principe général peux-tu dégager de cette expérience ? Comment l'appliquer ailleurs ou autrement ? Imagine comment réutiliser ce que tu apprends pour pouvoir faire des liens avec des situations futures.

5. Exemple

Dans ce chapitre, pour illustrer l'algorithme pour penser les algorithmes, nous vous présentons un exemple d'exercice résolu. Nous avons choisi l'exercice numéro 10101_2 : le Maximisateur. Ce 21^e exercice est extrait du manuel[2] à l'attention des enseignants qui accompagne le dispositif didactique débranché *human processor!.* L'énoncé est le suivant :

Pour chaque paire de nombres, recopie le plus grand des deux sur la sortie.

Il est donc proposé de :

- Lire une paire de nombres sur l'entrée.
- Déterminer quel est le plus grand des deux nombres.
- Recopier le plus grand nombre sur la sortie.
- Recommencer avec la prochaine paire de valeurs.

Un challenge accompagne cet exercice et précise que le tout peut être programmé en utilisant 10 instructions du jeu *human processor !* qui ne comprend qu'un ensemble réduit d'instructions. (Nous renvoyons le lecteur au manuel pour plus d'explications sur les différentes instructions) :

- Lire une valeur, depuis une entrée ou une mémoire.
- Écrire une valeur, vers une sortie ou une mémoire.
- Faire un saut conditionnel si le contenu de l'accumulateur est nul ou s'il est négatif.
- Faire un saut inconditionnel dans le flux des instructions.

La solution proposée dans le manuel (indiquée ici en pseudo-code) est la suivante :

0. Début du code.
1. Adresse [0] : Lire une valeur sur l'entrée et la placer dans l'accumulateur.
2. Écrire la valeur de l'accumulateur vers la mémoire.
3. Lire une autre valeur sur l'entrée et la placer dans l'accumulateur.
4. Soustraire la valeur en mémoire au contenu de l'accumulateur (N.B.: le résultat de la soustraction reste dans l'accumulateur).

[2] Le manuel est téléchargeable sur le site : http://human-processor.xyz/telechargement.php?doc=ManuelEnseignant

5. Si le résultat qui est dans l'accumulateur est négatif, sauter à l'adresse [2].
6. Sinon, ajouter la valeur en mémoire au contenu de l'accumulateur (N.B.: le résultat de l'addition reste dans l'accumulateur).
7. Sauter à l'adresse [1].
8. Adresse [2] : Lire la valeur qui est en mémoire et la placer dans l'accumulateur.
9. Adresse [1] : Écrire la valeur qui est dans l'accumulateur sur la sortie.
10. Recommencer (sauter à la l'adresse [0]).
11. Fin du code (jamais atteinte).

Voici un processus de résolution possible qui nous permet de trouver la solution optimale en 10 instructions seulement. Pour chaque étape de la réflexion, sont indiquées laquelle des trois phases de l'algorithme nous nous trouvons et la fonction cognitive mobilisée.

[1. Récolte] [Méthode - Rigueur]

Lire l'énoncé.

Pour chaque => correspond à une boucle infinie => mettre 1 JUMP inconditionnel en fin de code

Paire de valeurs => 2 instructions READ pour la première et la deuxième valeur

Recopie sur la sortie => 1 instruction WRITE du résultat sur la sortie

[2. Assemblage] [Évocation]

Synthèse de ce qui est demandé.

Total 4 instructions déjà utilisées. Indice : solvable en 10 instructions.

[3. Projection] [Reformulation]

Comment, en 6 instructions, identifier la plus grande valeur ?

[3. Projection] [Structuration]

Émettre une hypothèse sur la démarche qui conduira à la solution.

J'ai besoin de comparer deux valeurs => Je dois utiliser au moins une mémoire et l'accumulateur, voir deux mémoires, mais je ne sais pas comparer deux mémoires entre elles. (*Fin du cycle d'élaboration d'une image mentale du problème*).

[4a. Réflexion] [Inférence - Déduction]

Conceptualiser une manière possible de résoudre le problème. Visualiser l'état du modèle mental.

Je n'ai pas d'opérateur de comparaison de valeurs entre les mémoires, mais je peux par contre calculer la différence et regarder le signe du résultat pour les comparer. Si c'est positif ou négatif, je saurais laquelle des deux valeurs est la plus grande selon l'ordre de la soustraction.

[4b. Idée] [Pensée divergente]

Chercher à exploiter tout ce qu'il est possible de faire avec ces données.

Pour aller plus vite, je peux peut-être n'utiliser qu'une seule mémoire et, après avoir effectué la soustraction, soit inverser la soustraction pour retrouver la valeur précédente de l'accumulateur, soit retourner la première valeur qui aura été mémorisée. Cette idée ne m'est pas venue tout de suite. J'ai d'abord programmé une solution qui marche utilisant plus d'instructions que le minimum optimal et je suis ensuite reparti dans mes réflexions pour avoir cette idée.

[4a, Réflexion] [Induction - Intériorisation]

Déplacer les données avec les yeux. Créer de nouveaux états du modèle mental. Décrire les opérations effectuées en langage naturel.

Compte tenu de l'idée et de l'inférence ci-dessous, je mets la première valeur en mémoire => reste 5 instructions. Je lis la seconde valeur => instruction déjà décomptée lors de l'étape de récolte des données => reste 5 instructions. Puis je soustrais la première à la seconde valeur (qui est toujours dans l'accumulateur) => reste 4 instructions. Je teste si le résultat est négatif (je ne peux pas tester autre chose) => reste 3 instructions.

[5. Vérification] [Monitoring]

Est-ce que la solution est juste ?

À ce niveau, je ne sais pas quoi écrire dans mon test, car, du fait que je suis dyslexique, je n'arrive pas à visualiser le résultat de la soustraction et déterminer qui est le plus grand des deux. Je dois donc faire un effort de réflexion particulier pour décider laquelle des deux valeurs est la plus grande si le résultat de la soustraction est négatif.

[5. Vérification] [Récapitulation]

Ai-je vérifié que toutes les données du problème sont bien satisfaites ?

Mais une des instructions est (voir l'idée) d'ajouter la valeur en mémoire à l'accumulateur pour retrouver la deuxième valeur lue, sans avoir besoin de la mémoriser, dans le cas où c'est la deuxième qui est la plus grande. Dans l'autre cas, c'est la première valeur qui est la plus grande, il suffit de lire ce qui avait en mémoire. (*Fin du cycle d'élaboration d'une image mentale de la solution*).

[5. Vérification] [Précision]

Je code ce qui précède. Je fais attention à n'utiliser que trois instructions. Je vois qu'il ne manque qu'un saut à la fin du test pour que le code fonctionne correctement : c'est ma 10ème instruction.

[6. Expression] [Autocontrôle]

Après avoir codé, je place les dernières instructions et vérifie que le code fait bien ce qui est prévu.

--- Fin de la résolution du problème, début de la phase d'analyse réflexive[3] *---*

[7. Métacognition] [Introspection]

Pour rédiger cette analyse, je repasse au ralenti les différentes étapes de sa résolution.

[7. Métacognition] [Transferts]

Je me dis que cette analyse peut être étendue à tous les exercices et pourrait servir pour expliciter mon raisonnement aux élèves en difficulté. Je choisis ainsi de la rédiger pour ce problème en particulier.

[3] Par souci de concision, nous ne détaillons pas dans cet article les fonctions cognitives mobilisées lorsque le problème est résolu, c'est-à-dire dans la phase d'analyse réflexive ou dans la phase d'apprentissage. Nous renvoyons le lecteur intéressé par ces dimensions au manuel enseignant du dispositif didactique qui aborde rapidement ces questions dans le chapitre 11 (version juillet 2020).

6. Conclusion

Finalement, le meilleur moyen de développer l'intelligence algorithmique de l'élève, c'est de la mobiliser sur des problèmes simples au début pour alléger sa charge cognitive (Lucile Chanquoy, André Tricot, John Sweller, 2007) développer la précision de son modèle mental (Philip N. Johnson-Laird, 1983) et la puissance de sa mémoire de travail en activant ses neurones de manière répétée et espacée (Steve Masson, 2020), puis ensuite sur des problèmes variés de plus en plus complexes, pour qu'il puisse acquérir de nouvelles stratégies de résolution de problème et enrichir ses expériences. Mais dans tous les cas, il faut être capable d'accepter le doute de ne pas savoir faire, pour dépasser cet état et apprendre, petit à petit, à développer sa créativité, sa logique mathématique et son esprit pratique : les trois composantes fondamentales de l'intelligence algorithmique.

L'algorithme théorique de résolution de problèmes algorithmiques fait l'objet de travaux de recherche en didactique de l'informatique à l'Université de Patras (Grèce) sous la codirection du Professeur Vassilis Komis et à la Haute École Pédagogique de Lausanne (Suisse) sous la codirection du Professeur Bernard Baumberger.

L'aptitude à résoudre des problèmes algorithmiques chez l'élève dépendra de 5 paramètres :

1. Sa faculté à rester concentrer sur la tâche en cours, malgré les perturbations extérieures, afin de réussir à se construire une représentation mentale fidèle de l'état du système durant chaque étape de son processus de transformation, au fil de l'avancement de la réflexion.
2. L'étendue de sa mémoire de travail et le nombre d'objets mentaux qu'il sera à même de prendre en considération simultanément afin de se constituer un modèle mental fidèle du système avec lequel il doit travailler.
3. De bonnes hypothèses et stratégies cognitives de résolution de problèmes, avec un riche panel d'expériences et de stratégies ayant eu du succès auparavant. Cette richesse ne s'acquiert qu'avec l'expérience. Il faut donc s'entrainer à concevoir des algorithmes pour développer cette faculté.
4. Son aptitude à gérer ses doutes et ses émotions lorsqu'il sera bloqué et qu'il ne saura plus quoi faire. Sa capacité à créer un état de tranquillité

intérieure, en situation de stress, afin de laisser au cerveau la possibilité de trouver de nouvelles idées face au problème rencontré. Sa faculté à cultiver sa créativité en situation de blocage en acceptant sereinement de ne pas savoir quoi faire, sans paniquer, sans douter, sans se soucier du fait de ne pas savoir.

5. La puissance du plaisir narcissique qu'il ressentira en résolvant les premiers problèmes et qui déterminera sa motivation à explorer davantage de problèmes algorithmiques.

Ainsi, un élève qui manquera de confiance dans sa capacité d'apprendre à résoudre des problèmes algorithmiques, se retrouvera en difficulté non pas parce qu'il ne peut pas apprendre à les résoudre, mais par ce qu'il ne supportera pas l'état de doute et d'incertitude nécessaire pour traverser la phase où l'on ne sait ni quoi faire ni comment faire.

Une autre cause possible à son manque de persévérance est son sentiment d'impuissance perçue (Yann Le Bossé, 2016) : "à quoi bon faire l'effort de continuer d'essayer si de toute façon je perds mon temps, car je n'en suis pas capable". Il faut accepter avec humilité le fait que l'acte intellectuel qui conduit à la maitrise de l'intelligence algorithmique n'est pas inné, mais qu'il s'acquiert, avec du temps, des efforts et en utilisant de bonnes stratégies.

Nous avons présenté dans cet article un ensemble de 16 stratégies susceptibles d'aider les élèves en difficulté à traverser sereinement les moments où l'on ne sait pas faire, jusqu'à ce que, avec l'expérience positive des premiers exercices, l'élève développe un modèle mental efficient du fonctionnement de l'ordinateur et commence à développer son intelligence algorithmique jusqu'à ce qu'il puisse sortir, un peu plus rapidement à chaque fois, de ces moments où il ne sait pas faire.

Nous pensons que l'explicitation de fonctions et de stratégies cognitives en classe est une aide précieuse pour l'enseignement de l'intelligence algorithmique au plus grand nombre. Être capable d'identifier les blocages potentiels chez les élèves en difficulté demande une connaissance fine des mécanismes cognitifs à l'œuvre dans la dynamique d'apprentissage de l'algorithmique. L'algorithme de résolution de problèmes algorithmiques est une aide, à la fois pour l'enseignant et pour les élèves, qui permet d'avoir une vision d'ensemble des processus cognitifs en œuvre lors de la conception d'une solution algorithmique à un problème posé. Son apprentissage permet de prendre conscience de la complexité de l'acte de résolution de problèmes

algorithmiques et des difficultés qui y sont liées. Un travail similaire, moins spécifique à l'enseignement de l'informatique, pourrait également être mené sur les processus cognitifs mobilisés dans l'acte d'apprendre. Le lecteur intéressé trouvera quelques pistes dans le livret de cours qui accompagne le dispositif didactique *human processor!*.

L'étude du développement de l'intelligence algorithmique, c'est-à-dire des intelligences créative, logico-mathématique et pratique des élèves, est en soi un vaste domaine. En classe, il est nécessaire de se préoccuper de l'expérience vécue par l'élève dans son ensemble : de ne pas se limiter à la chose à apprendre, mais d'étendre notre enseignement au développement de l'affect et de toutes les intelligences de l'élève, quelle que soit la discipline enseignée, y compris dans l'acte même d'apprendre. Ainsi, nous pensons qu'il faudrait compléter la formation des enseignants par l'étude approfondie des mécanismes de développement des intelligences de l'élève. Nous nous devons d'étudier l'acte de compréhension de l'élève via la perception, l'imagination et la mémoire : c'est-à-dire étudier le *noos* (dans l'œuvre de Platon « intelligence, esprit en tant qu'il perçoit et qu'il pense ») ou la *noèse* (dans la pensée phénoménologique d'Edmund Husserl (Leclercq et Richard, 2016) « l'activité du sujet connaissant »). Pour ce faire, nous proposons d'envisager la création d'une nouvelle discipline, complémentaire à la didactique et la pédagogie (voir figure 4 : une revisite du triangle pédagogique), fondée sur la psychologie de la créativité, la psychologie cognitive et les neurosciences de l'éducation, dont l'objet d'étude serait le développement des intelligences de l'élève à l'école et que nous pourrions nommer : la *noèsiologie*[4].

[4] Nous avons créé ce terme suite à une étude étymologique des concepts sous-jacents. Il n'y a pas de corrélation avec le terme *noesiologie* (sans accent sur le "*e*") utilisé sur le site https://maiminana.com ou https://www.noesiology.com. Leur définition est « La science qui étudie les effets de la pensée sur la vie ». Notre définition est « La science qui étudie le développement de l'intelligence ».

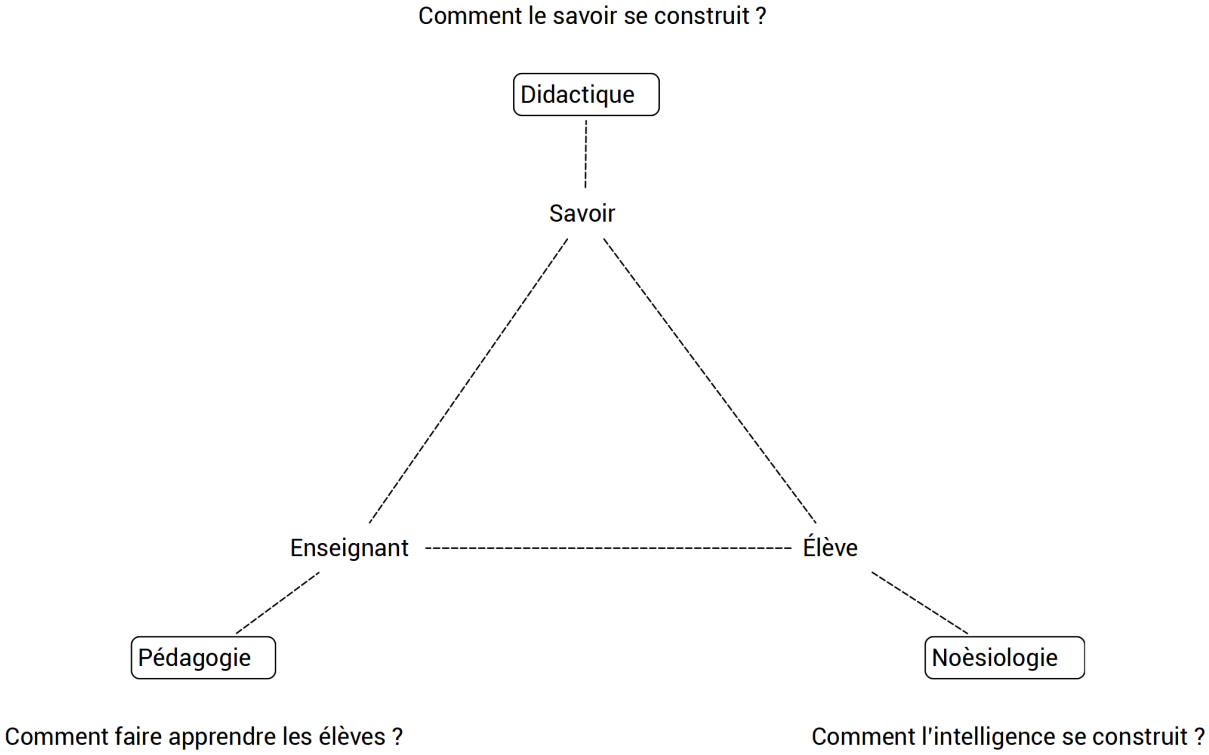


Figure 4. Une revisite du triangle pédagogique.

6. Bibliographie

Blanvillain, C. (2020). *Human Processor!* Dans les Actes du 3e colloque scientifique Ludovia, à Yverdon-les-Bains, Suisse, 6 – 8 avril. En ligne : https://www.ludovia.ch/2020/wp-content/uploads/2020/09/act_coll_scient_ludovia.ch_2020-V6.pdf

Chanquoy, L. Tricot, A. Sweller, J. (2007). *La charge cognitivie. Théorie et applications*. Paris : Armand Colin.

Chappaz, G. (1996). Comprendre et construire la médiation. Dans *Les médiations éducatives*, no 17
(pp. 7-24). Lille : Spirale. En ligne : https://www.persee.fr/doc/spira_0994-3722_1996_num_17_1

Fournier, J-Y. (1999). *A l'école de l'intelligence.* Paris : ESF.

Gabler, K. Blomquist, A. Gray, K. (2015). *Human Ressource Machine*. Tomorrow Corporation. En ligne : https://tomorrowcorporation.com/humanresourcemachine

Johnson-Laird, P. N. (1983). *Mental Models.* Cambridge, MA: Harvard University Press.

Le Bossé, Y. (2016). *Sortir de l'impuissance. Tome 2 : aspects pratiques.* Québec : Ardis.

Leclercq, B. Richard, S. (2016). « Husserl (A) ». Dans M. Kristanek, *l'Encyclopédie philosophique*. En ligne : https://encyclo-philo.fr/husserl-a

Lemaire, P. Didierjean, A. (2018). *Introduction à la psychologie cognitive.* Bruxelles : De Boeck Supérieur.

Lubart, T. Mouchiroud, C. Tordjman, S. Zenasni, F. (2015) *Psychologie de la créativité.* Paris : Armand Colin.

Lubart, T. Zenasni, F. Barbot, B. (2016). Le potentiel créatif : de la mesure à son développement. Dans
I. Capron Puozzo, *La créativité en éducation et formation* (pp. 65-78). Bruxelles : De Boeck Supérieur.

Madnick, S. (1965). *Little Man Computer*. Illinois State University. En ligne : https://fr.wikipedia.org/wiki/Little_man_computer

Masson, S. (2020). *Activer ses neurones : pour mieux apprendre et enseigner.* Paris : Odile Jacob.

Mayer, C. (2005). *Manuel de métapédagogie.* Menoncourt : Upbraining. En ligne : https://www.upbraining.net/produit/manuel-de-metapedagogie

Sternberg, R. J. (1985). *Beyond I.Q.: A triarchic theory of human intelligence.* New York: Cambridge University Press.

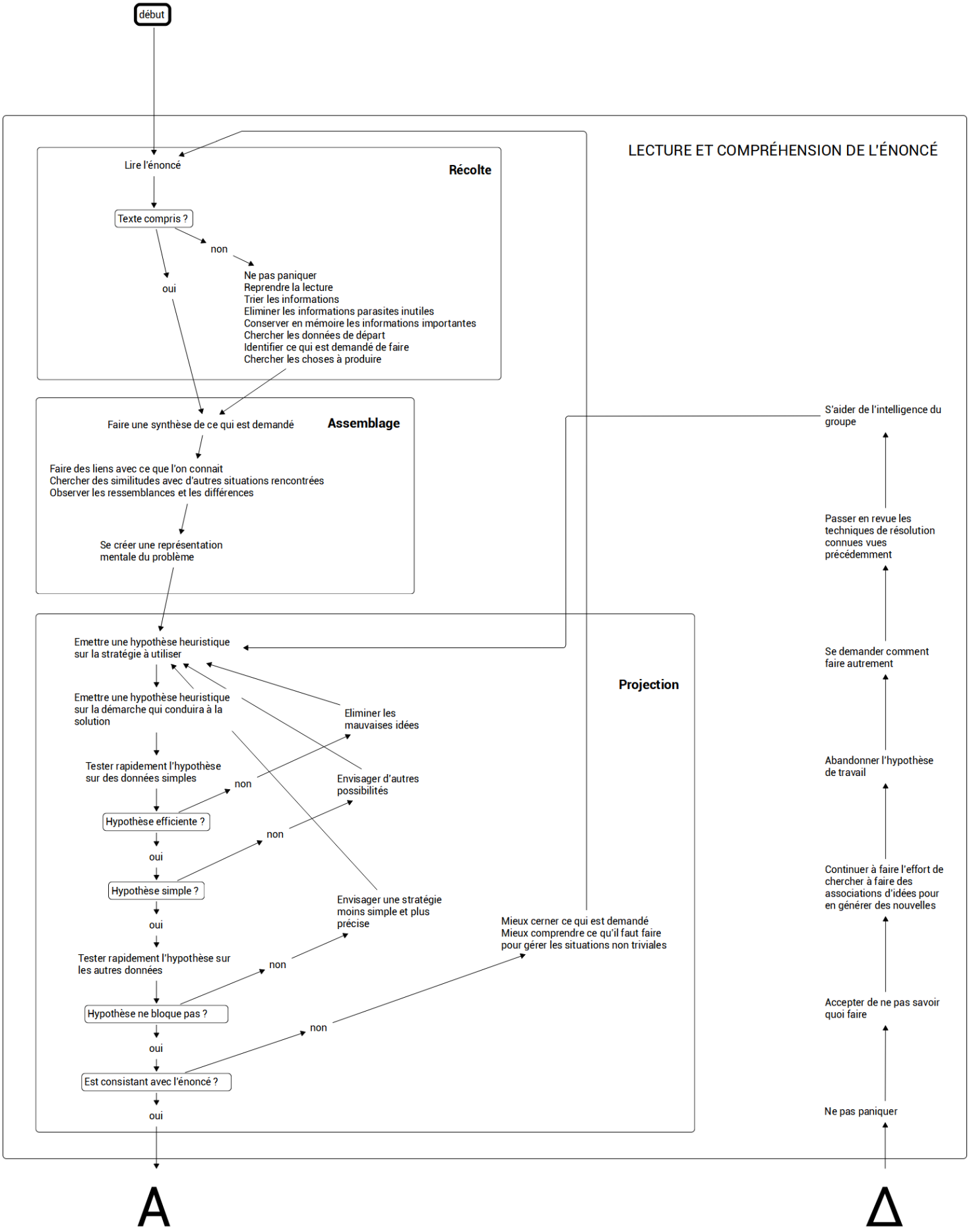
Vianin, P. (2009). *L'aide stratégique aux élèves en difficulté scolaire – Comment donner à l'élève les clés de sa réussite ?* Bruxelles : De Boeck.

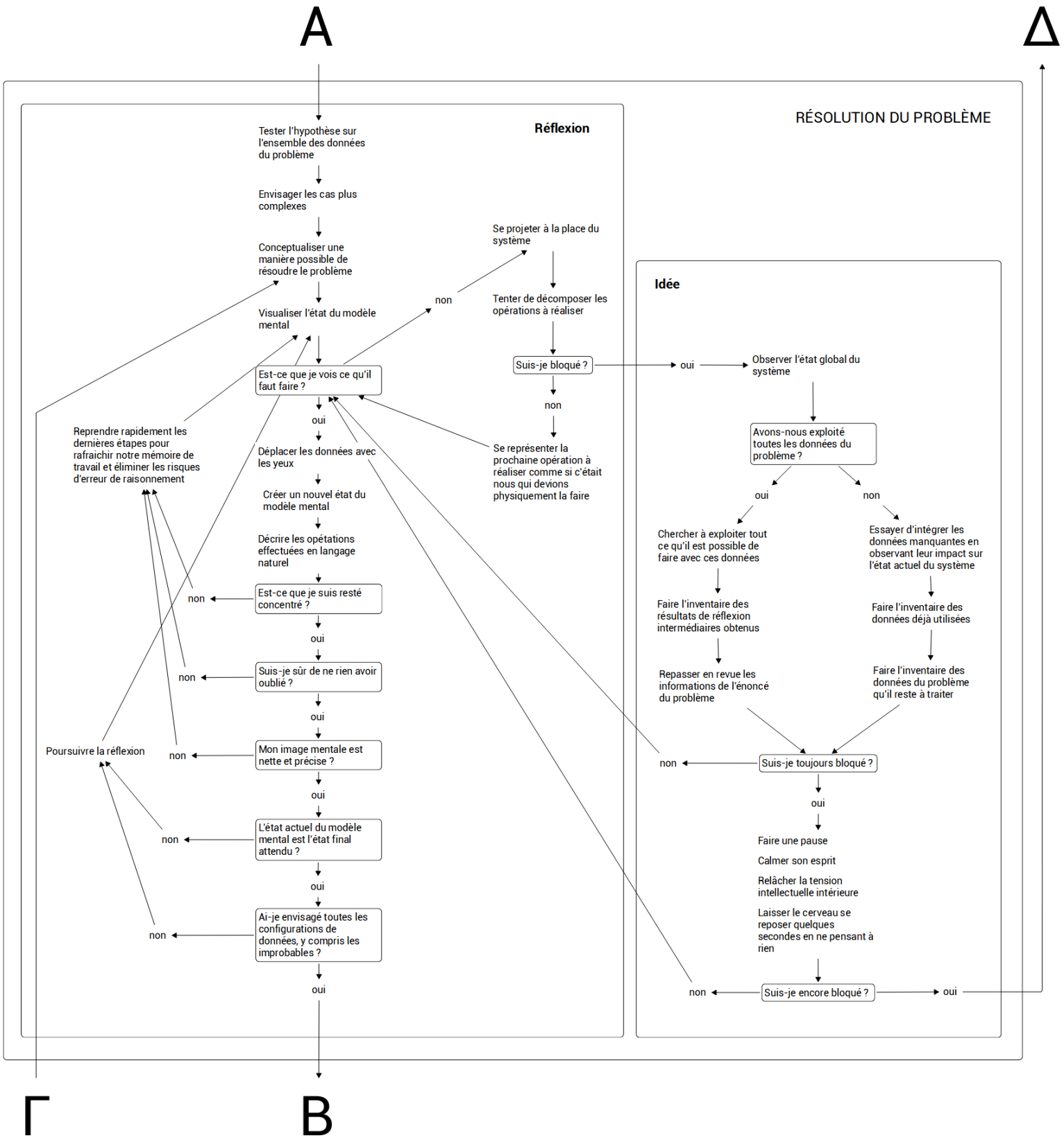
Vianin, P. (2010). *Neurosciences cognitives et pédagogie spécialisée : un exemple d'évaluation diagnostique des processus cognitifs.* Berne : Centre Suisse de Pédagogie Spécialisée. En ligne : https://edudoc.ch/record/102518

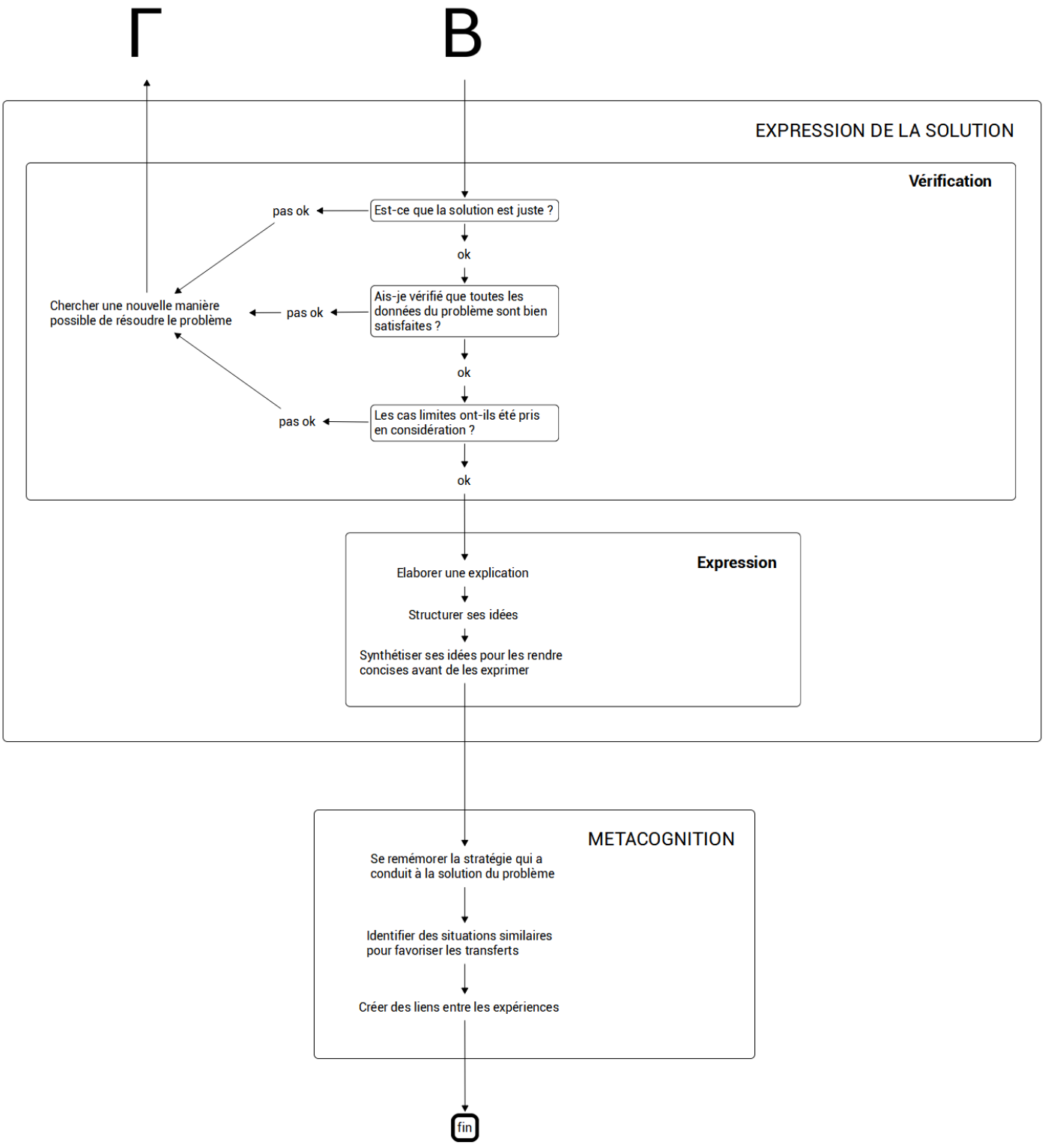
7. Annexe

Les trois prochaines pages d'annexe présentent un diagramme de synthèse qui reprend les principales idées de l'algorithme pour apprendre à penser les algorithmes, objet du chapitre 2.

Les textes dessinés dans une bulle représentent des questions et correspondent à des étapes conditionnelles dans lesquelles un branchement est effectué selon que la réponse à la question posée est positive ou négative. Les grosses lettres grecques représentent des connexions entre les trois parties de l'algorithme.


début
LECTURE ET COMPRÉHENSION DE L'ÉNONCÉ
Lire l'énoncé
Récolte
Texte compris ?
non
oui
Ne pas paniquer
Reprendre la lecture
Trier les informations
Eliminer les informations parasites inutiles
Conserver en mémoire les informations importantes
Chercher les données de départ
Identifier ce qui est demandé de faire
Chercher les choses à produire
Faire une synthèse de ce qui est demandé
Assemblage
Faire des liens avec ce que l'on connait
Chercher des similitudes avec d'autres situations rencontrées
Observer les ressemblances et les différences
Se créer une représentation mentale du problème
Emettre une hypothèse heuristique sur la stratégie à utiliser
Projection
Emettre une hypothèse heuristique sur la démarche qui conduira à la solution
Eliminer les mauvaises idées
Tester rapidement l'hypothèse sur des données simples
non
Envisager d'autres possibilités
Hypothèse efficiente ?
non
oui
Hypothèse simple ?
Envisager une stratégie moins simple et plus précise
oui
Tester rapidement l'hypothèse sur les autres données
non
Hypothèse ne bloque pas ?
non
oui
Est consistant avec l'énoncé ?
oui
Mieux cerner ce qui est demandé
Mieux comprendre ce qu'il faut faire
pour gérer les situations non triviales
S'aider de l'intelligence du groupe
Passer en revue les techniques de résolution connues vues précédemment
Se demander comment faire autrement
Abandonner l'hypothèse de travail
Continuer à faire l'effort de chercher à faire des associations d'idées pour en générer des nouvelles
Accepter de ne pas savoir quoi faire
Ne pas paniquer
A
Δ


A
Δ
RÉSOLUTION DU PROBLÈME
Réflexion
Idée
Tester l'hypothèse sur l'ensemble des données du problème
Envisager les cas plus complexes
Conceptualiser une manière possible de résoudre le problème
Visualiser l'état du modèle mental
Est-ce que je vois ce qu'il faut faire ?
non
Se projeter à la place du système
Tenter de décomposer les opérations à réaliser
Suis-je bloqué ?
oui
non
Se représenter la prochaine opération à réaliser comme si c'était nous qui devions physiquement la faire
Observer l'état global du système
Avons-nous exploité toutes les données du problème ?
oui
non
Chercher à exploiter tout ce qu'il est possible de faire avec ces données
Faire l'inventaire des résultats de réflexion intermédiaires obtenus
Repasser en revue les informations de l'énoncé du problème
Essayer d'intégrer les données manquantes en observant leur impact sur l'état actuel du système
Faire l'inventaire des données déjà utilisées
Faire l'inventaire des données du problème qu'il reste à traiter
Suis-je toujours bloqué ?
non
oui
Faire une pause
Calmer son esprit
Relâcher la tension intellectuelle intérieure
Laisser le cerveau se reposer quelques secondes en ne pensant à rien
Suis-je encore bloqué ?
non
oui
Reprendre rapidement les dernières étapes pour rafraichir notre mémoire de travail et éliminer les risques d'erreur de raisonnement
Déplacer les données avec les yeux
Créer un nouvel état du modèle mental
Décrire les opétations effectuées en langage naturel
Est-ce que je suis resté concentré ?
non
oui
Suis-je sûr de ne rien avoir oublié ?
non
oui
Mon image mentale est nette et précise ?
non
oui
L'état actuel du modèle mental est l'état final attendu ?
non
oui
Ai-je envisagé toutes les configurations de données, y compris les improbables ?
non
oui
Poursuivre la réflexion
Γ
B


Г
B
EXPRESSION DE LA SOLUTION
Vérification
Est-ce que la solution est juste ?
pas ok
ok
Ais-je vérifié que toutes les données du problème sont bien satisfaites ?
pas ok
Chercher une nouvelle manière possible de résoudre le problème
ok
Les cas limites ont-ils été pris en considération ?
pas ok
ok
Expression
Elaborer une explication
Structurer ses idées
Synthétiser ses idées pour les rendre concises avant de les exprimer
METACOGNITION
Se remémorer la stratégie qui a conduit à la solution du problème
Identifier des situations similaires pour favoriser les transferts
Créer des liens entre les expériences
fin